Lawrence Livermore National Laboratory

Lustre at Scale
The LLNL Way

D. Marc Stearman

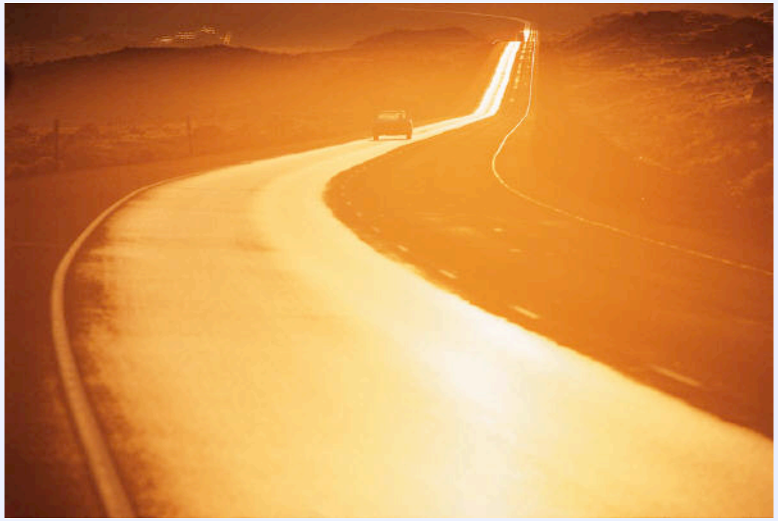
Lustre Administration Lead
Livermore Computing - LLNL

This work performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.

LLNL-PRES-403067

Topics

- Project Structure
- LLNL Computing Platforms
- Network Design and Topology
- Software Release Methodology
- Operation and Management
- Next Steps
- Hyperion
- Concerns

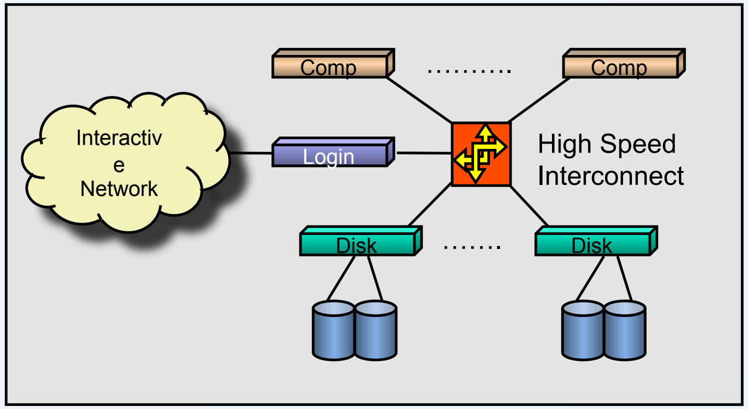


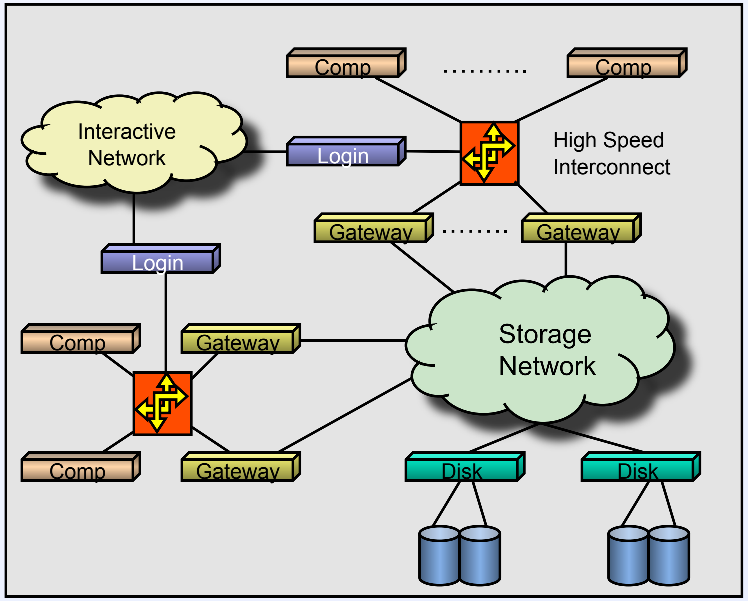
Project Structure - "Who are all these people, and what are they doing here?"

- Project Lead - Mark Gary
- Administration/Operations Team - Marc Stearman
  - Responsible for daily system administration, cluster integration, upgrades, hardware repair, user application support
  - 4 Full Time Employees + HW Repair Team
- Software Development - Jim Garlick
  - Responsible for bug fixing, build, QA, and tool development
  - 4 Full Time Employees

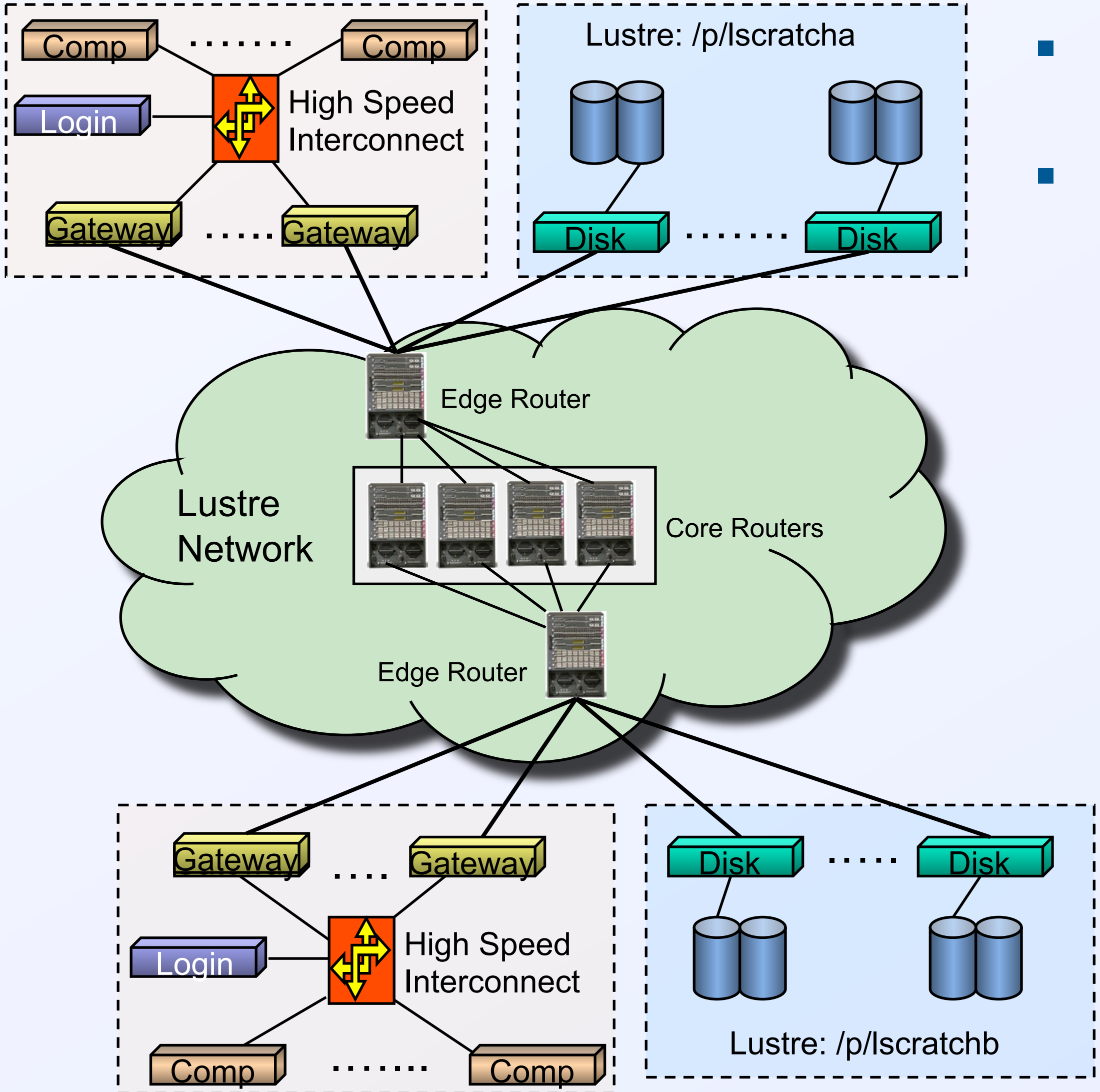
HPC Compute/Storage Pairing Philosophies

- Islands of Storage
  - Many HPC clusters, with dedicated attached storage
  - Data access internal to cluster
  - High performance - Uses local high-speed interconnect
  - Many copies of data running on multiple clusters
- Peninsulas of Computation
  - Many HPC clusters using shared network storage
  - User convenience - Fewer copies of data
  - Redundant - if one file system is down, others are still available
  - Extra network latency



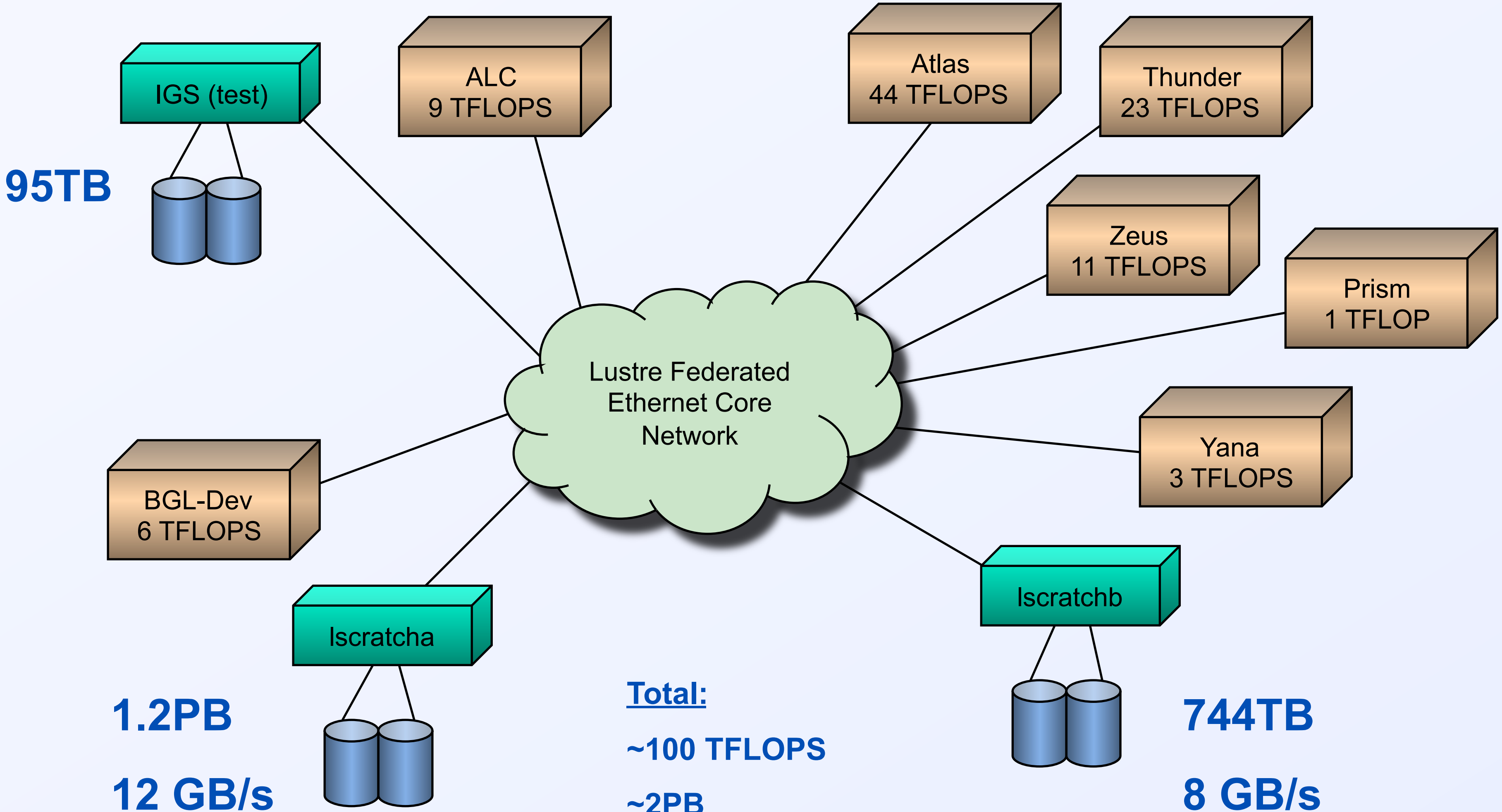


Peninsulas Explained



- All storage visible via storage network
- Compute clusters have some Lustre file systems more "local" than others

Livermore Computing - Open Computing Facility

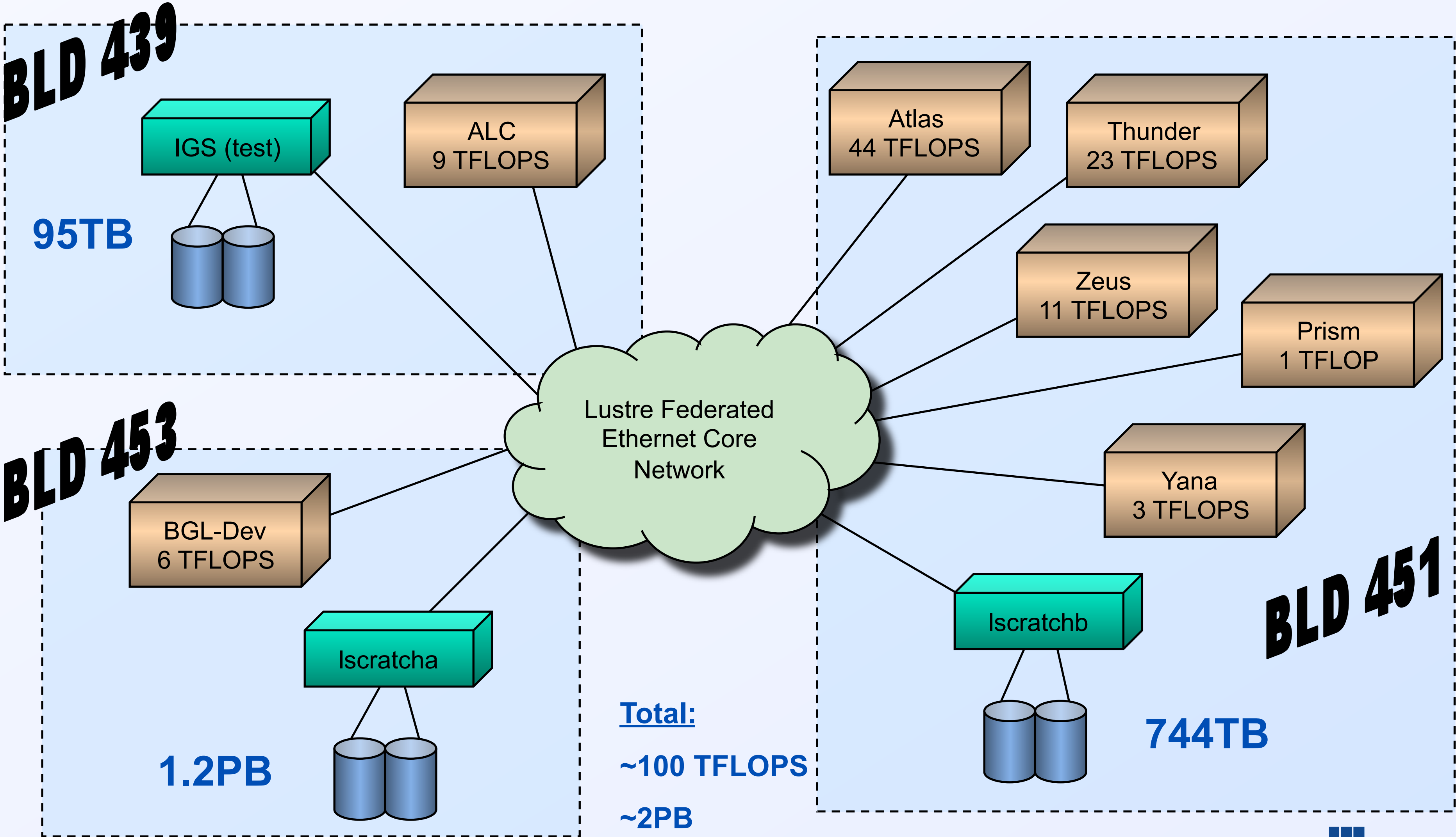


Total:

~100 TFLOPS

~2PB

Livermore Computing - Open Computing Facility

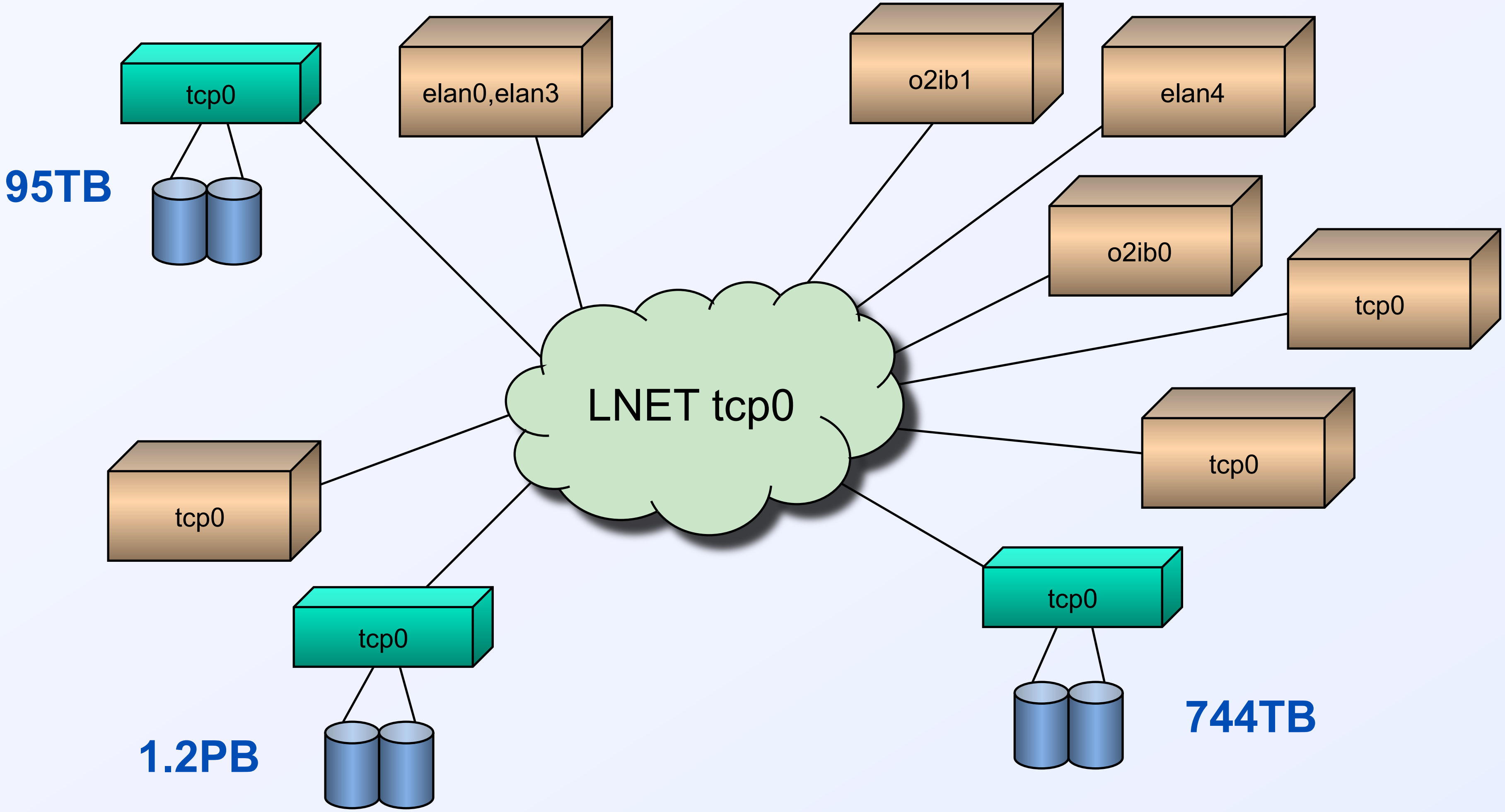


Total:

~100 TFLOPS

~2PB

LNET view of the world - Open Computing Facility

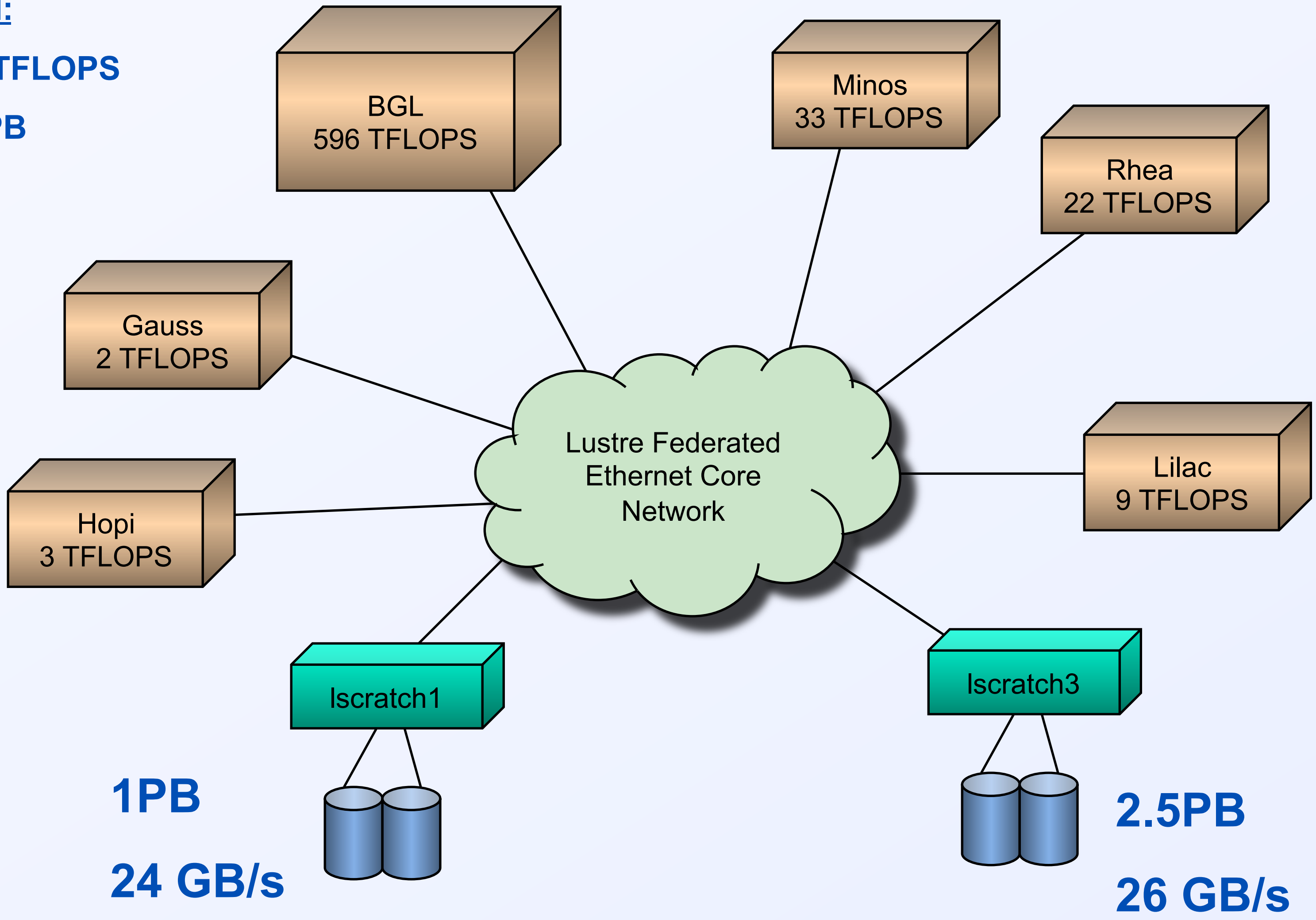


Livermore Computing - Secure Computing Facility

Total:

665 TFLOPS

3.5 PB

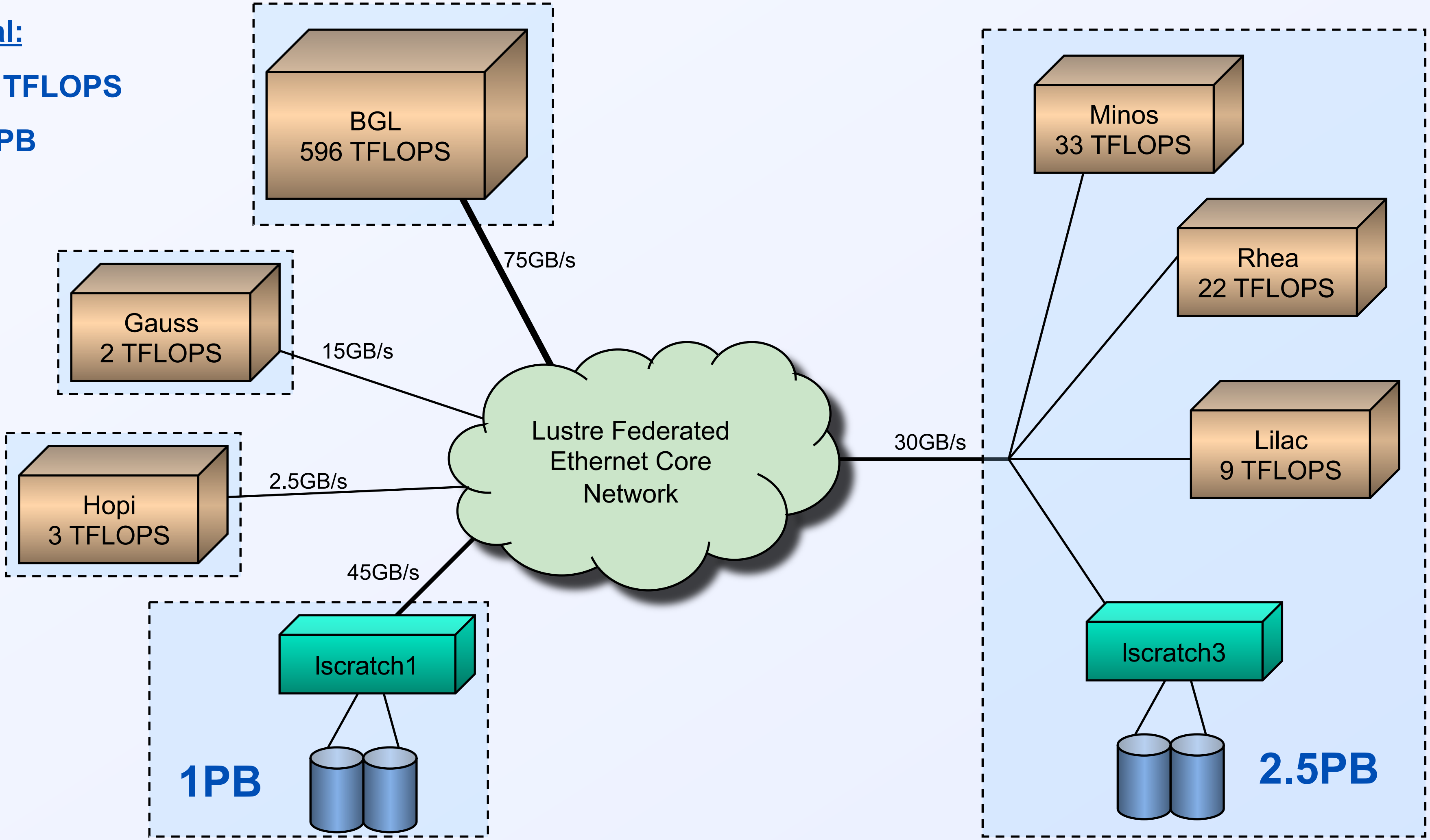


Livermore Computing - Secure Computing Facility

Total:

665 TFLOPS

3.5 PB

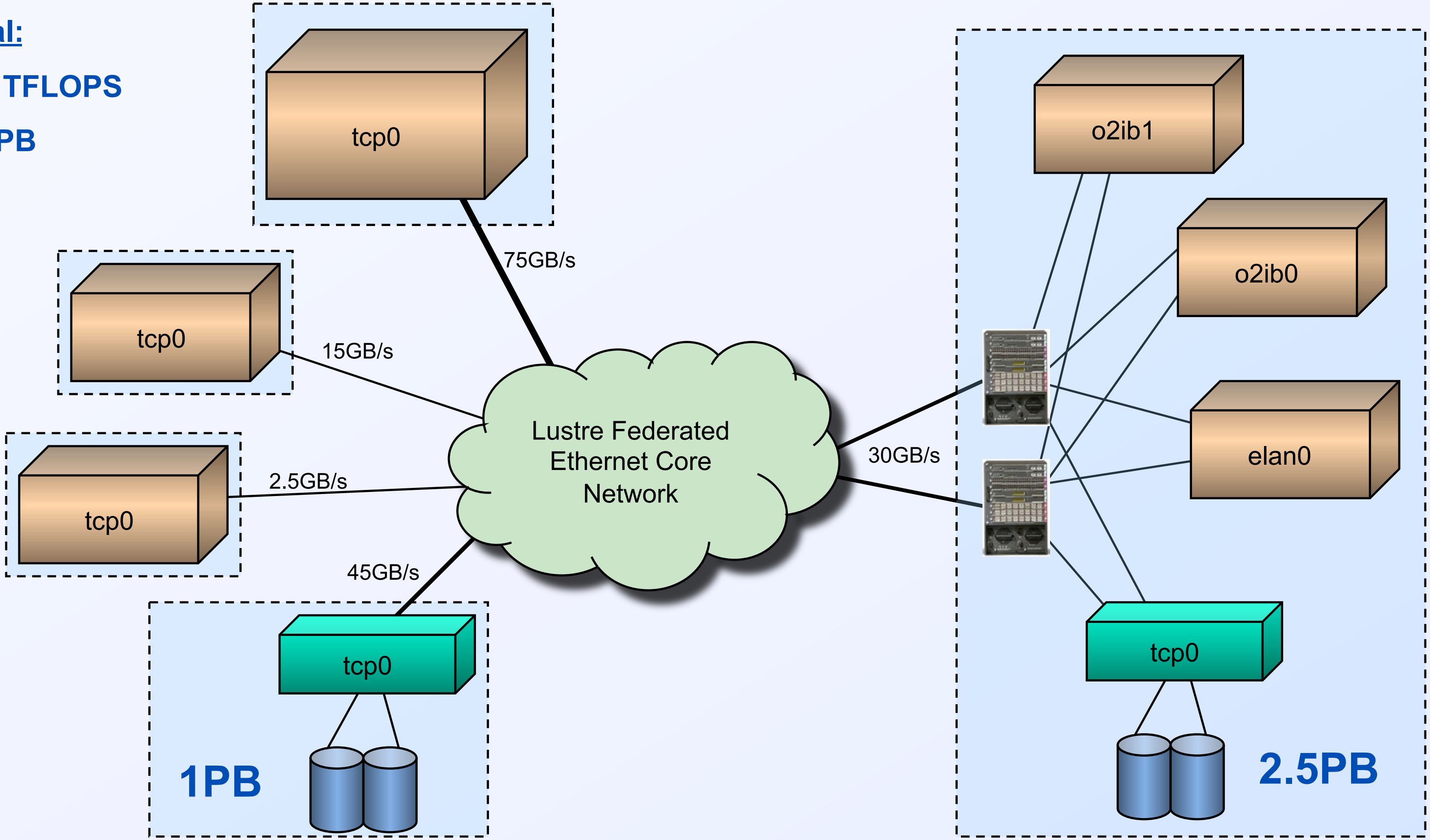


LNET view of the world - Secure Computing Facility

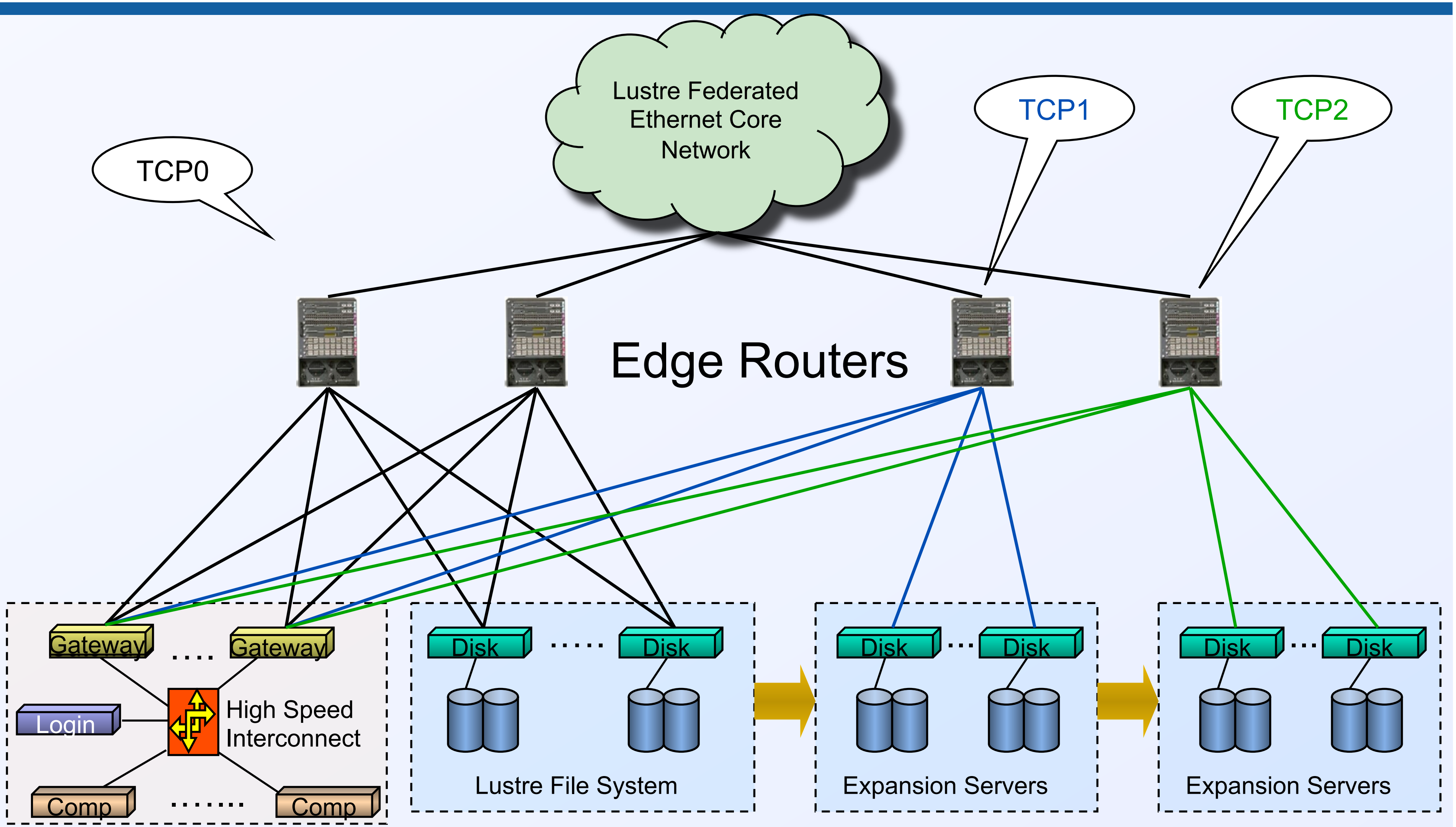
Total:

665 TFLOPS

3.5 PB



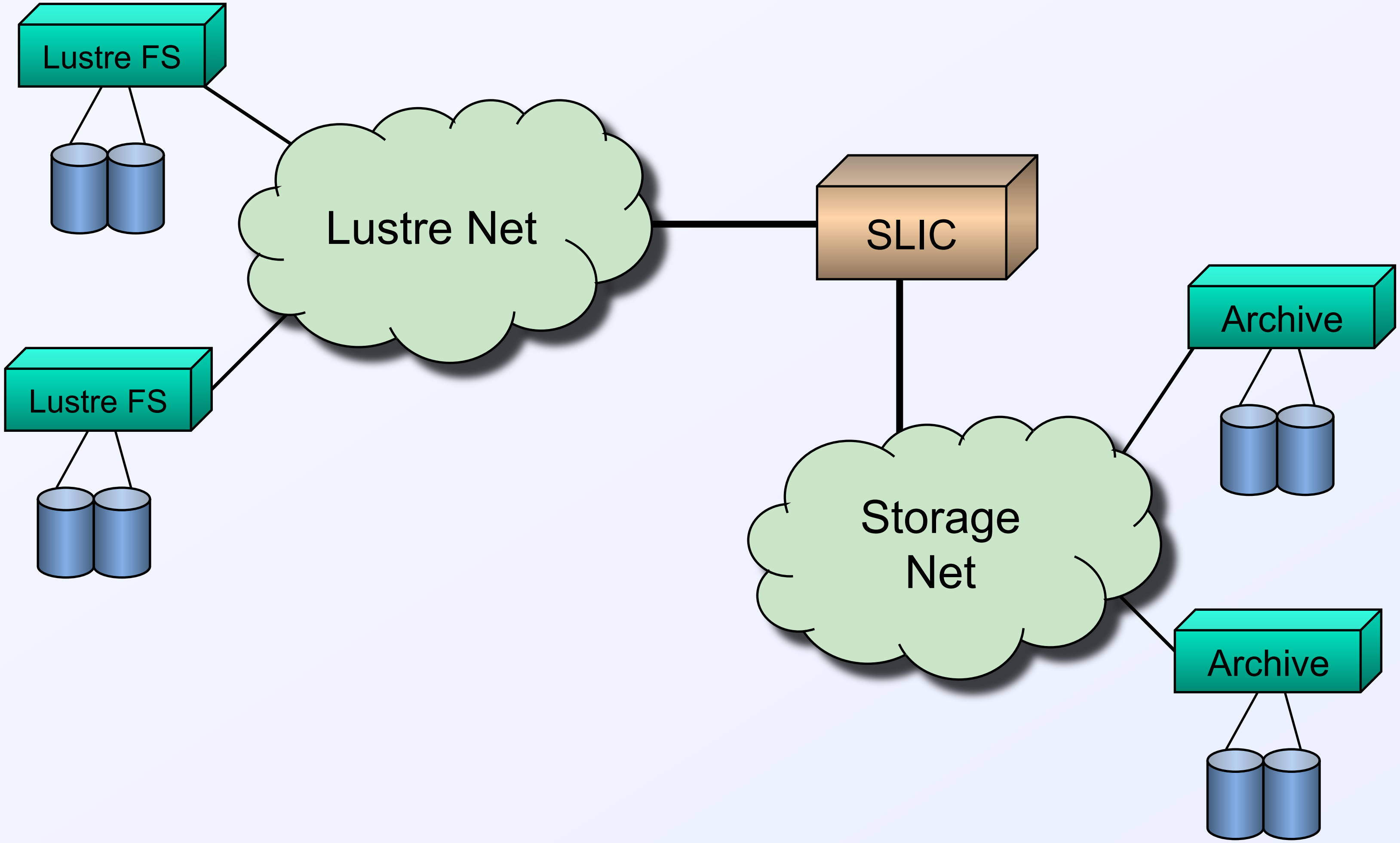
Livermore Computing - Next Steps (LNET)



Livermore Computing - Next Steps (LNET)

- Benefits
  - Reduced network traffic through the core switches
  - Keeps traffic local to switch backplanes and reducess latency
  - Saves Money!
- Issues
  - Need dynamic NIDs
    - Routes can be added dynamically via ‘lctl’ on cluster with routers, but tcp-only clients on tcp0 need unload of lustre modules to add tcp1 and tcp2 NIDs
    - With multiple lustre filesystems mounted, ALL filesystems must be unmounted so all jobs are killed limiting production work
  - Makes LNET configuration more complex (But we already have that)

SLIC - Storage Lustre Interface Cluster



Software Release - Testing Methodology

- Sanity test in Build Farm
- Small scale testing in Testbed
- Mid scale testing on BGL-Dev, and new 64 node cluster
- Large scale testing on ALC (~400 clients)
- Giant scale testing on Atlas (1100 clients) during DST (Dedicated System Time)
- Wide variety of tests - IOR, iozone, fsx, MIB, mdtest, simul, PIOS, various reproducers
- Developers are using Lustre for their /home file system
- We carry 150+ patches from base releases to make lustre work on our production systems - This requires a great deal of testing resources

Software Release - Rollout Methodology

- Midway through migration from 1.4 to 1.6
- Servers before Clients
  - All servers now running 1.6.2-30chaos (1.6.2 + 160 patches)
  - All x86_64 clients and BGL moving to 1.6.2-30chaos
  - All i686 and ia64 clients will remain at 1.4.8-22.3chaos (1.4.8 + 156 patches) until they retire (about 6-9 months from now)
- The 1.4 clients can mount the 1.6 file systems because they were created under 1.4, and migrated to 1.6
- If we field a new file system we have three options for our legacy clients
  - Create it at 1.4 and migrate to 1.6
  - Develop a tool that will read the 1.6 on-disk configuration, and write out 1.4 style config
  - Not mount legacy clients

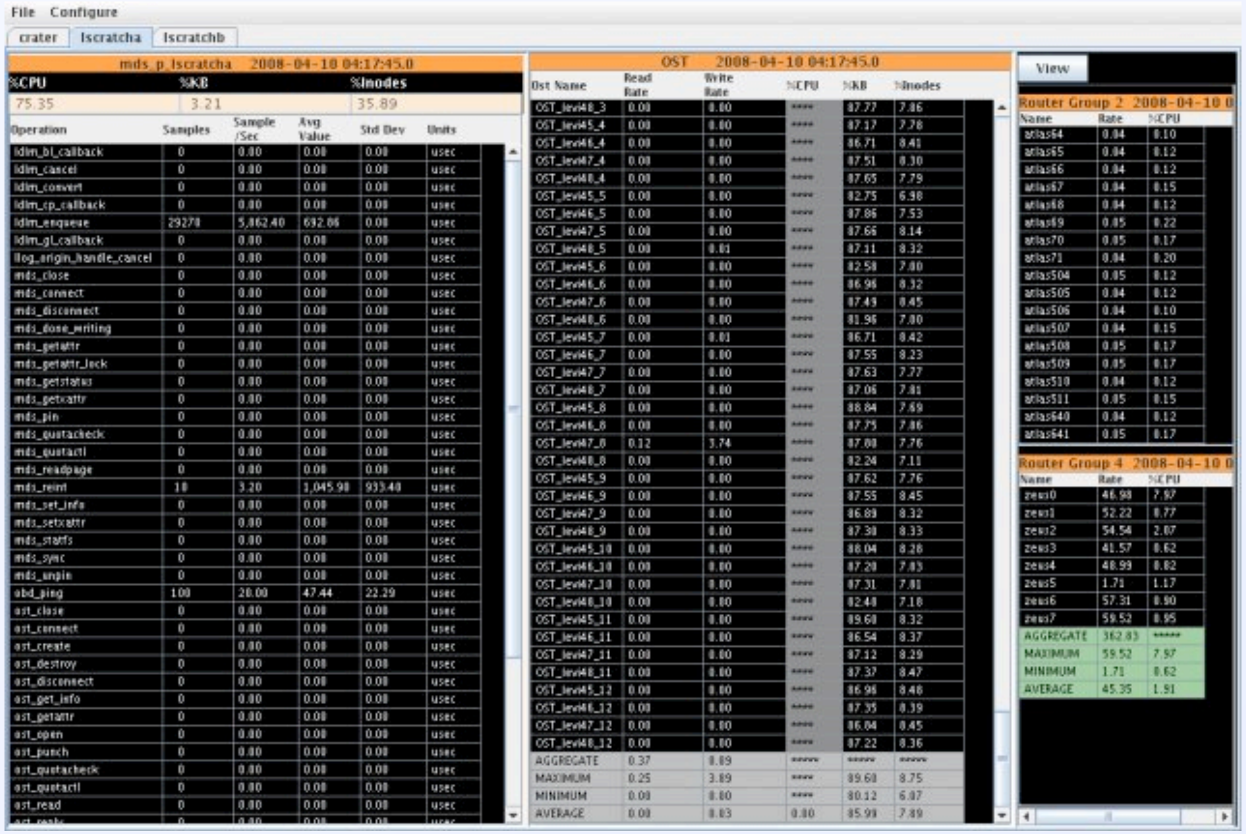
Operations and Management - "Sameness"

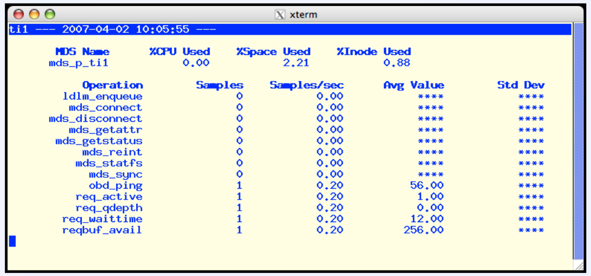
- Storage Scalable Units (SSUs)
  - Concept borrowed from Compute Clusters
  - Building blocks used to build/expand file systems
  - Makes deployment smooth and quick
- Hardware Repair team
  - Operations Personnel handle all HW repair actions with minimal SysAdmin intervention
- Operator training
  - Training courses developed locally
  - Knowledge Base on local wiki
  - Goal to reduce off-hours pages
  - Numerous scripts for testing and problem determination
- LMT v2

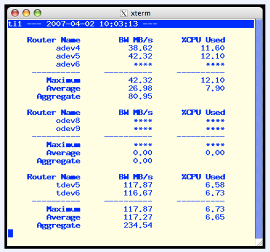
LMT v2 - http://sourceforge.net/projects/lmt/

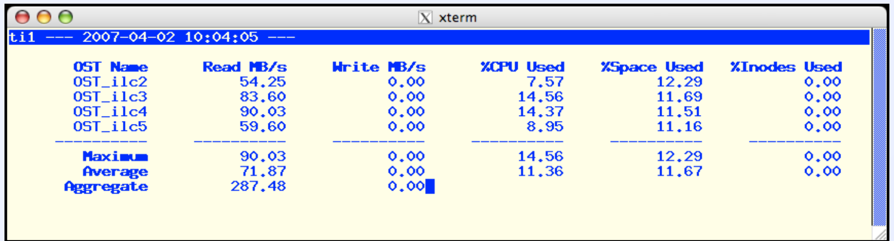
Start with xwatch-lustre functionality, then add:

- New views (OSS, Filesystem, Router Group, …)
- Plotting capability (historical trends, heart-beat, …)
- Customization features
- Full-system health "at a glance"



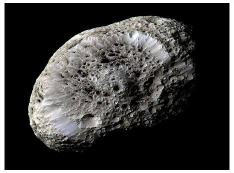






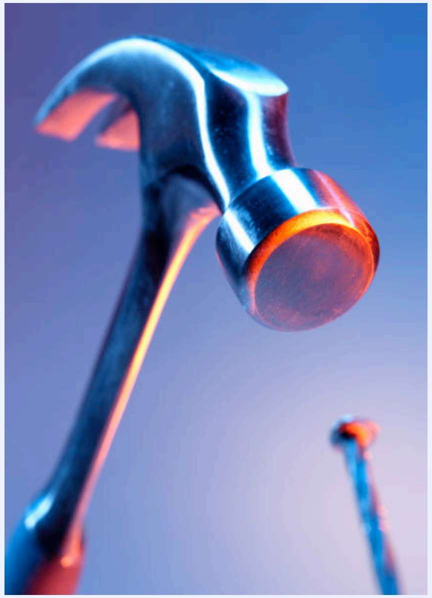
Livermore Computing - Next Steps

- Filesystem Requirements
  - Dawn (.5 PFLOPS): 96 GB/s, ~4PB
  - Sequoia (10 PFLOPS): 512 GB/s, ~50PB
- Router Cluster
  - Sequoia may have an Infiniband network
  - Need to bridge Infiniband with legacy 10GE Net
- Evaluating ZFS (Both User Space and In Kernel)
- Failover



Hyperion

- Will replace ALC as large scale lustre test cluster
- Hyperion is a partnership with the vendor community, including Sun/CFS
- Will test and evaluate emerging technologies - OFED, Virtualization, Lustre, QDR Infiniband, 40/100 GE, etc
- 1152 nodes, 9K cores, 120 TFLOPS
- 1.6 PB disk @ >36 GB/s
- Two Lustre Networks
  - Infiniband
  - 10 Gigabit Ethernet



Concerns

- Space Management
  - How do you manage a 50 PB file system?
  - Quotas?
  - Purges?
  - Conventional tools do not scale
  - ls, tar, cp, rsync, etc
  - Moving data to Archive efficiently
- ZFS
  - Performance: User space vs. Kernel space
  - Migration from ldiskfs to ZFS
- Scaling concerns
  - Metadata Performance
  - Adaptive Timeouts
  - Multi/Many core parallelization
- Seemless Failover

Questions